

Universidade da Beira Interior

Departamento de Informática



**Departamento de
Informática**

Dice And Tables

Elaborado por:

João Miguel Sena Baptista Carneiro

Orientador:

Mestre Nuno Carapito

24 de junho de 2025

Agradecimentos

A conclusão deste trabalho, bem como da grande maior parte da minha vida académica, não seria possível sem a ajuda de várias pessoas, às quais deixo o meu mais sincero obrigado.

Em primeiro lugar, gostaria de agradecer ao meu professor orientador, Nuno Carapito, cuja orientação e contribuições foram fundamentais para a realização deste projeto. Sem o seu apoio e conhecimento, este trabalho não teria sido possível.

Agradeço também aos meus professores pela transmissão de conhecimento e pela disponibilidade para esclarecer dúvidas ao longo de todo o percurso académico.

À minha família, deixo o meu agradecimento pela paciência nos momentos mais difíceis e pelo apoio constante ao longo destes anos.

Por fim, agradeço aos meus amigos e colegas, que estiveram sempre presentes para partilhar alegrias e frustrações, assim como momentos de descanso e de estudo. A companhia deles foi essencial para tornar o percurso mais leve e com maior significado.

Por fim, a todas as pessoas que, de forma direta ou indireta, contribuíram para o meu crescimento pessoal e académico, deixo uma palavra de gratidão.

Conteúdo

Conteúdo	iii
Lista de Figuras	vii
Lista de Tabelas	ix
Lista de Excertos de Código	xi
1 Introdução	1
1.1 Enquadramento e Motivação	1
1.2 Objetivos	2
1.3 Organização do Documento	3
1.4 Disponibilização do Projeto	3
2 Tecnologias Utilizadas e Soluções Existentes	5
2.1 Introdução	5
2.2 Soluções Existentes	5
2.3 Tecnologias e Bibliotecas Utilizadas	7
2.3.1 Tecnologias e Bibliotecas Utilizadas no <i>Back-End</i>	7
2.3.1.1 <i>JavaScript</i>	7
2.3.1.2 <i>Node.js</i>	7
2.3.1.3 <i>Express</i>	7
2.3.1.4 <i>Sequelize</i>	7
2.3.1.5 <i>MariaDB</i>	8
2.3.1.6 <i>SQLite3</i>	8
2.3.1.7 <i>Sequelize CLI</i>	8
2.3.1.8 <i>bcrypt</i>	8
2.3.1.9 <i>jsonwebtoken</i>	8
2.3.1.10 <i>cookie-parser</i>	8
2.3.1.11 <i>cors</i>	9
2.3.1.12 <i>dotenv</i>	9
2.3.1.13 <i>multer</i>	9
2.3.1.14 <i>fs</i>	9

2.3.1.15	<i>Stripe</i>	9
2.3.1.16	<i>pdfkit</i>	9
2.3.1.17	<i>qrcode</i>	9
2.3.1.18	<i>nodemailer</i>	9
2.3.1.19	<i>node-cron</i>	10
2.3.1.20	<i>swagger-jsdoc</i> e <i>swagger-ui-express</i>	10
2.3.2	Tecnologias e Bibliotecas Utilizadas no <i>Front-End</i> . . .	10
2.3.2.1	HTML e CSS	10
2.3.2.2	<i>React</i>	10
2.3.2.3	<i>TanStack React Query</i>	11
2.3.2.4	<i>Axios</i>	11
2.3.2.5	<i>js-cookie</i>	11
2.3.2.6	<i>React Toastify</i>	11
2.3.2.7	<i>@stripe/stripe-js</i>	11
2.4	Ferramentas de Apoio ao Desenvolvimento	11
2.4.1	<i>Visual Studio Code (VSCode)</i>	11
2.4.2	<i>Insomnia</i>	12
2.4.3	<i>GitHub</i>	12
2.4.4	<i>HeidiSQL</i>	12
2.5	Conclusões	12
3	Análise de Requisitos e Modelação da Base de Dados	13
3.1	Introdução	13
3.2	Requisitos Funcionais	13
3.3	Requisitos Não Funcionais	14
3.4	Diagrama da Base de Dados	15
3.4.1	Estrutura e Relacionamento das Tabelas	16
3.5	Conclusões	16
4	Implementação e Testes	17
4.1	Introdução	17
4.2	Funcionalidades do Utilizador	17
4.2.1	Registo e <i>Autenticação</i>	18
4.2.2	Gestão do Perfil	19
4.2.3	Procurar Cafés	20
4.2.4	Reservas de Mesas e Jogos	20
4.2.5	Juntar-se a Grupos de Reservas	21
4.2.6	Compras de Jogos	22
4.3	Funcionalidades de Gestão e Administração	23
4.3.1	Gestão de Cafés	23
4.3.2	Gestão de Mesas	24

4.3.3	Gestão de Jogos	25
4.3.4	Gestão de Reservas	26
4.3.5	Gestão de Utilizadores e Cargos (Administração)	27
4.4	Implementação Técnica do <i>Back-End</i> e <i>Front-End</i>	28
4.4.1	<i>Back-End</i>	28
4.4.2	<i>Front-End</i>	30
4.5	Testes e Validação	31
4.5.1	Testes de Funcionalidade	31
4.5.2	Testes de Integração	32
4.5.3	Validação Manual	33
4.6	Conclusões	33
5	Conclusões, Dificuldades Encontradas e Trabalho Futuro	35
5.1	Principais Conclusões	35
5.2	Dificuldades Encontradas	35
5.3	Trabalho Futuro	36
	Bibliografia	39

Lista de Figuras

1.1	Prazos Estabelecidos para as Tarefas do Semestre	2
3.1	Diagrama da Base de Dados criado no <i>DBSchema</i>	15
4.1	Formulário de Registo de Novo Utilizador	18
4.2	Área de Gestão de Perfil do Utilizador	19
4.3	Lista de Cafés Disponíveis na Plataforma	20
4.4	Criação de Nova Reserva	21
4.5	Opção para Juntar-se a um Grupo de Reserva	22
4.6	Interface de Compra de Jogos	22
4.7	Painel de Gestão do Café	23
4.8	Interface de Gestão de Mesas Disponíveis	24
4.9	Interface de Gestão de Jogos Disponíveis	25
4.10	Interface de Gestão de Reservas	26
4.11	Interface de Administração de Utilizadores e Cargos	27
4.12	Documentação automática da API com Swagger	30

Lista de Tabelas

2.1	Comparação entre plataformas e o sistema <i>Dice & Tables</i>	6
3.1	Requisitos Funcionais do Sistema	14
3.2	Requisitos Não Funcionais do Sistema	14
4.1	Validação dos Requisitos Funcionais	32

Lista de Excertos de Código

4.1	Envio de fatura com QR Code via email	29
4.2	Tarefa agendada para atualizar reservas e stock	29
4.3	Criação de sessão de pagamento com Stripe	29

Acrónimos

2FA	Autenticação de Dois Fatores
API	<i>Application Programming Interface</i>
HTML	<i>HyperText Markup Language</i>
CSS	<i>Cascading Style Sheets</i>
CORS	<i>Cross-Origin Resource Sharing</i>
HTTP	<i>Hypertext Transfer Protocol</i>
JWT	<i>JSON Web Token</i>
ORM	<i>Object-Relational Mapping</i>
PDF	<i>Portable Document Format</i>
REST	<i>Representational State Transfer</i>
CRUD	<i>Create, Read, Update and Delete</i>

Capítulo

1

Introdução

1.1 Enquadramento e Motivação

Nos últimos anos tem-se visto um grande crescimento de interesse por jogos de tabuleiro[1], não só como forma de entretenimento, mas também como uma atividade social cada vez mais valorizada. Com isso, têm vindo a abrir cada vez mais cafés temáticos dedicados a este *hobby*[2], bem como outros estabelecimentos que, mesmo não tendo um stock próprio, estão disponíveis para acolher grupos de jogadores com regularidade. Contudo, para os amantes e simpatizantes deste hobby, continua a ser difícil encontrar cafés que aceitem receber estes jogadores, saber se estes têm mesas disponíveis ou até formar grupos para jogar.

Esta dificuldade reflete a ausência de uma plataforma centralizada que permita aos jogadores pesquisar cafés, verificar a disponibilidade de mesas, reservar jogos e organizar partidas. Da mesma forma, os próprios cafés sentem a necessidade de uma ferramenta que facilite a gestão de reservas, mesas e stock de jogos.

É neste contexto que surge a ideia de desenvolver uma plataforma digital que junte jogadores e cafés num único espaço online, com uma interface simples e funcional. Este projeto foi desenvolvido com uma preocupação na organização e estrutura da aplicação, abrangendo tanto o *Front-End* como o *Back-End*, utilizando tecnologias modernas como *React*[3] e *Node.js*[4].

Para além de contribuir para uma melhor experiência dos jogadores e uma gestão mais eficiente por parte dos cafés de jogos de tabuleiro, este projeto representa também uma oportunidade de crescimento pessoal, ao permitir o contacto direto com ferramentas e metodologias de desenvolvimento web que são altamente relevantes no mercado atual.

1.2 Objetivos

O principal objetivo deste projeto é o desenvolvimento de uma plataforma para Gestão de Cafés de Jogos de Tabuleiro, com a implementação de um servidor *Back-End* e de uma interface *Front-End* intuitiva e funcional.

Com isto em mente, o desenvolvimento foi dividido em seis tarefas:

1. Contextualização com os objetivos propostos; preparação do ambiente de trabalho, seleção e familiarização com as tecnologias a utilizar
2. Implementação do Servidor *Back-end* em *Node.js*
3. Implementação da Plataforma em *React*
4. Ligação do *Front-End* ao *Back-End*
5. Finalização do projeto. Potencial implementação de novas funcionalidades. Testes e melhorias
6. Escrita do relatório de projeto

Foram também estabelecidos prazos para cada uma das tarefas a realizar durante o semestre, como demonstra a Figura 1.1.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
T1															
T2															
T3															
T4															
T5															
T6															

Figura 1.1: Prazos Estabelecidos para as Tarefas do Semestre

Assim sendo, pretende-se que a aplicação desenvolvida possa ser acedida online através de um domínio próprio. Também deve permitir aos utilizadores registarem-se, visualizar cafés com jogos, verificar disponibilidade e realizar reservas. Além disso, é possível comprar jogos, formar grupos de jogo, receber notificações por email e aceder a funcionalidades de gestão específicas para cafés e administradores.

1.3 Organização do Documento

De modo a refletir o trabalho desenvolvido, este documento encontra-se estruturado da seguinte forma:

1. O **Capítulo 1 – Introdução** – apresenta o projeto, o seu enquadramento e motivação, os objetivos definidos e a organização do documento.
2. O **Capítulo 2 – Tecnologias e Ferramentas Utilizadas** – descreve as tecnologias, bibliotecas e ferramentas aplicadas no desenvolvimento do sistema, tanto no *Back-End* como no *Front-End*, incluindo também os ambientes de suporte ao desenvolvimento.
3. O **Capítulo 3 – Análise de Requisitos e Modelação da Base de Dados** – detalha os requisitos funcionais e não funcionais do sistema, assim como a estrutura e os relacionamentos da base de dados.
4. O **Capítulo 4 – Implementação e Testes** – descreve as principais funcionalidades implementadas, tanto do ponto de vista do utilizador como da administração, abordando também a estrutura técnica da aplicação e os testes realizados para garantir o seu bom funcionamento.
5. O **Capítulo 5 – Conclusões e Trabalho Futuro** – apresenta as principais conclusões retiradas com o desenvolvimento do projeto, bem como possíveis melhorias e extensões futuras.

1.4 Disponibilização do Projeto

Com o objetivo de promover a transparência, reprodutibilidade e eventual colaboração futura, o projeto desenvolvido encontra-se disponível publicamente. O código-fonte pode ser consultado no repositório oficial do GitHub, enquanto a aplicação está acessível numa versão funcional online.

- **Repositório GitHub:** <https://github.com/JoaoCarneiroo/Dice-Tables>
- **Plataforma Online:** <https://diceandtables.pt>

Tecnologias Utilizadas e Soluções Existentes

2.1 Introdução

Este capítulo apresenta as tecnologias e as ferramentas utilizadas durante o desenvolvimento do projeto, bem como os seus respectivos propósitos. Antes disso, é feita uma análise das soluções já existentes no mercado, com o objetivo de contextualizar a necessidade e pertinência da plataforma.

O conteúdo está dividido em três secções principais:

- **Soluções Existentes** 2.2, onde é feito um levantamento de algumas iniciativas atuais relacionadas com o contexto dos jogos de tabuleiro e a sua limitação em cafés e espaços sociais;
- **Tecnologias e Bibliotecas Utilizadas** 2.3, onde são descritas as linguagens de programação, *frameworks* e bibliotecas que foram utilizadas para a implementação do sistema;
- **Ferramentas de Apoio ao Desenvolvimento** 2.4, onde se enumeram os softwares e plataformas usados ao longo do projeto para facilitar o desenvolvimento.

2.2 Soluções Existentes

O crescente interesse pelos jogos de tabuleiro nas últimas décadas tem conduzido à criação de diversas iniciativas e plataformas que procuram aproximar jogadores e espaços físicos dedicados à prática deste passatempo. Al-

gumas plataformas, como o *BoardGameGeek* [5], centram-se na partilha de conteúdos, classificações e críticas sobre jogos, enquanto outras, como o *Meetup* [6], facilitam a organização de encontros sociais, incluindo sessões de jogos de tabuleiro.

Apesar destas iniciativas, são muito poucas as soluções verdadeiramente orientadas para a realidade dos cafés temáticos ou estabelecimentos que acolhem jogadores com regularidade. A maioria das plataformas existentes não disponibiliza funcionalidades específicas, como a verificação de disponibilidade de mesas, a reserva de jogos ou a formação de grupos de jogadores.

Existem plataformas generalistas de reservas, como o *OpenTable* [7], focadas no setor da restauração, mas que não contemplam as necessidades específicas dos cafés de jogos de tabuleiro. Algumas soluções pontuais, como o *BookingNinja* [8] ou projetos open-source como o “Board-Game-Cafe-Reservation-System” [9], têm sido desenvolvidas por cafés ou comunidades locais, mas revelam-se frequentemente limitadas em termos de escalabilidade, manutenção ou usabilidade.

A Tabela 2.1 resume as principais funcionalidades disponibilizadas por diferentes plataformas e destaca os diferenciais da solução proposta, *Dice & Tables*, face às restantes.

Tabela 2.1: Comparação entre plataformas e o sistema *Dice & Tables*

Funcionalidade	BoardGameGeek	MeetUp	OpenTable	Dice & Tables
Pesquisa de cafés com jogos disponíveis	Não	Não	Sim	Sim
Consulta de stock de jogos	Sim	Não	Não	Sim
Formação de grupos para jogar	Não	Sim	Não	Sim
Reserva de mesas com gestão de horários	Não	Não	Sim	Sim
Gestão de reservas com integração de jogos	Não	Não	Não	Sim
Compra de jogos com faturação automática	Não	Não	Não	Sim
Pagamentos online integrados	Não	Não	Sim	Sim
Envio de emails automáticos	Não	Não	Sim	Sim
Geração de QR Codes	Não	Não	Não	Sim

2.3 Tecnologias e Bibliotecas Utilizadas

Esta secção apresenta as tecnologias e ferramentas utilizadas no desenvolvimento do projeto, divididas em duas categorias principais: *Back-End* (2.3.1) e *Front-End* (2.3.2). Cada secção descreve as bibliotecas, *frameworks* e ferramentas que desempenharam um papel importante na construção da plataforma.

2.3.1 Tecnologias e Bibliotecas Utilizadas no *Back-End*

2.3.1.1 *JavaScript*

Linguagem usada para implementar toda a lógica do *Back-End*, incluindo o tratamento de pedidos, autenticação e manipulação dos dados da aplicação.

2.3.1.2 *Node.js*

Node.js é um motor de execução de *JavaScript* utilizado para desenvolver o servidor da aplicação, suportando a gestão das rotas, a comunicação com a base de dados e a integração de várias bibliotecas. A sua natureza assíncrona e baseada em eventos permitiu criar uma aplicação escalável, com tempos de resposta reduzidos, ideal para sistemas com múltiplos acessos simultâneos.

2.3.1.3 *Express*

Framework minimalista e flexível para *Node.js* que fornece um robusto conjunto de funcionalidades para a aplicação web e Application Programming Interface (*API*) Representational State Transfer (*REST*) [10]. Constitui a base do servidor HTTP da aplicação. A sua sintaxe simples e estrutura modular facilitam a criação de rotas e *middlewares*, acelerando o desenvolvimento e organização do código.

2.3.1.4 *Sequelize*

Object-Relational Mapping (*ORM*) que facilita a comunicação entre o código *JavaScript* e bases de dados [11]. Permite definir modelos e realizar operações na base de dados de forma abstrata e estruturada. Esta abordagem permitiu garantir a integridade dos dados e facilitou a manutenção do esquema da base de dados ao longo do desenvolvimento.

2.3.1.5 *MariaDB*

Biblioteca utilizada para estabelecer a ligação com a base de dados *MariaDB*, que funciona como o sistema principal de armazenamento de dados da aplicação [12].

2.3.1.6 *SQLite3*

Sistema de base de dados leve baseado em ficheiros [13]. Foi utilizado em ambiente de desenvolvimento e testes para as interações com uma base de dados local.

2.3.1.7 *Sequelize CLI*

Ferramenta de linha de comandos para o *Sequelize*. Permite criar modelos, migrações e *seeders* de forma automática e estruturada, facilitando o controlo de versões do esquema da base de dados. A sua utilização garantiu consistência entre ambientes de desenvolvimento e produção.

2.3.1.8 *bcrypt*

Biblioteca utilizada para cifrar a palavra-passe de forma segura, protegendo os dados dos utilizadores [14]. Utiliza um algoritmo de *hashing* robusto, com *salting*, o que dificulta ataques por força bruta aumentando a segurança geral do sistema.

2.3.1.9 *jsonwebtoken*

Utilizada para a criação e verificação de *JSON Web Token* (JWT), permitindo a autenticação e autorização de utilizadores no sistema [15]. Através deste mecanismo, é possível garantir que os utilizadores autenticados tenham acesso a certas rotas ou funcionalidades da aplicação, sem necessidade de sessões persistentes. Os JWT transportam informações codificadas sobre o utilizador, como o seu identificador e permissões, e podem ser verificados no servidor sem a necessidade de armazenamento no lado do servidor.

2.3.1.10 *cookie-parser*

Facilita a leitura e interpretação de *cookies* Hypertext Transfer Protocol (*HTTP*) presentes nos pedidos do cliente. Embora seja possível aceder aos *cookies* manualmente via cabeçalhos, esta biblioteca simplifica o seu tratamento, tornando mais eficiente a gestão de sessões e tokens de autenticação.

2.3.1.11 cors

Middleware que permite configurar as políticas de Cross-Origin Resource Sharing (CORS), fundamentais para permitir que o *Front-End* aceda à *API* de forma segura.

2.3.1.12 dotenv

Realiza a gestão de variáveis de ambiente através de um ficheiro `.env`, permitindo separar configurações sensíveis do código-fonte.

2.3.1.13 multer

Middleware para gestão de uploads de ficheiros enviados em formulários multipart/form-data, utilizado para envio de imagens.

2.3.1.14 fs

Módulo utilizado para manipulação do sistema de ficheiros, permitindo operações como a leitura e escrita de ficheiros no servidor.

2.3.1.15 Stripe

Biblioteca para a integração de pagamentos online, usada para processar transações de forma segura e integrada na aplicação [16].

2.3.1.16 pdfkit

Permite a geração de documentos em formato *Portable Document Format* (PDF) diretamente do *Back-End*, utilizado para a criação de faturas.

2.3.1.17 qrcode

Biblioteca que possibilita a geração de códigos QR, utilizado para a identificação da compra de um Jogo.

2.3.1.18 nodemailer

Utilizada para o envio de *emails* ao notificar da criação de uma conta, para envio de código de Autenticação de Dois Fatores (2FA) e também para notificação de uma compra junto com a fatura. Esta integração contribuiu para uma experiência de utilizador mais completa e profissional, reforçando a comunicação entre o sistema e os utilizadores.

2.3.1.19 *node-cron*

Permite o agendamento de tarefas no *Back-End* utilizando a sintaxe *cron*. Foi utilizado para executar, em intervalos regulares, um programa responsável por verificar se existem reservas expiradas com base na hora atual. Quando detetadas, essas reservas são removidas e o stock de jogos é atualizado automaticamente. Esta abordagem permite automatizar operações recorrentes sem necessidade de intervenção manual, mantendo o sistema atualizado em tempo real.

2.3.1.20 *swagger-jsdoc e swagger-ui-express*

Bibliotecas que permitem criar documentação *Swagger* a partir dos comentários *JSDoc* e integrar a interface gráfica do *Swagger* diretamente numa aplicação *Express*, proporcionando uma forma simples e interativa de explorar e testar os *endpoints* da API.

2.3.2 Tecnologias e Bibliotecas Utilizadas no *Front-End*

2.3.2.1 HTML e CSS

O *HyperText Markup Language* (HTML) é a linguagem base utilizada para estruturar o conteúdo das páginas web, definindo elementos como títulos, parágrafos, botões ou formulários. Já o *Cascading Style Sheets* (CSS) é utilizado para definir a aparência desses elementos, controlando aspectos como cores, margens, tamanhos ou disposição no ecrã. No contexto deste projeto, o HTML foi usado em conjunto com bibliotecas como o *React* para estruturar os componentes da interface, enquanto o CSS — com o apoio da *framework Tailwind* [17] — permitiu criar um design responsivo, limpo e moderno, adaptado a diferentes dispositivos.

2.3.2.2 *React*

React foi a biblioteca principal utilizada para a construção da interface da aplicação. Baseada em componentes reutilizáveis, permite criar interfaces de utilizador dinâmicas e interativas de forma modular, promovendo uma melhor organização e manutenção do código. Além disso, a vasta comunidade e o ecossistema de ferramentas associadas a *React* contribuíram para acelerar o desenvolvimento e garantir boas práticas [3].

2.3.2.3 *TanStack React Query*

TanStack React Query foi utilizado para simplificar a gestão de dados assíncronos, como o carregamento e atualização de cafés, reservas e jogos a partir da API. Esta biblioteca permite sincronizar automaticamente o estado da interface com os dados remotos, facilitando operações como o fetching, caching e atualização de dados [18]. A sua utilização contribuiu para uma melhor performance e uma experiência de utilizador mais fluida, reduzindo chamadas desnecessárias e mantendo a interface atualizada em tempo real.

2.3.2.4 *Axios*

Cliente HTTP utilizado para comunicar com o *Back-End* de forma simples, com controlo total sobre pedidos e respostas [19].

2.3.2.5 *js-cookie*

Biblioteca para leitura e escrita de *cookies* no *browser*, usada para guardar o *token* de autenticação do utilizador [20].

2.3.2.6 *React Toastify*

Biblioteca usada para apresentar notificações visuais ao utilizador, como mensagens de sucesso, de erro ou informação [21].

2.3.2.7 *@stripe/stripe-js*

Biblioteca oficial da *Stripe*, utilizada na integração do sistema de pagamentos online da aplicação [22].

2.4 Ferramentas de Apoio ao Desenvolvimento

Para além das bibliotecas e *frameworks* utilizadas na implementação do projeto, recorreram-se também a diversas ferramentas que apoiaram o processo de desenvolvimento, teste e gestão do código.

2.4.1 *Visual Studio Code (VSCode)*

Editor de código que foi utilizado para desenvolver tanto o *Front-End* como o *Back-End* da aplicação. que permite organizar os ficheiros do projeto e trabalhar com o repositório *Git* de forma integrada [23].

2.4.2 *Insomnia*

Aplicação utilizada para testar os *endpoints* da API desenvolvida no *Back-End*. Facilitou o envio de pedidos HTTP com diferentes métodos, permitindo verificar de forma rápida as respostas do servidor [24].

2.4.3 *GitHub*

Aplicação utilizada para alojar o repositório do projeto. Facilitou o controlo de versões, a sincronização do código entre dispositivos e a gestão do histórico de alterações através do *Git* [25].

2.4.4 *HeidiSQL*

Ferramenta gráfica utilizada para visualizar e interagir com a base de dados *MariaDB* do projeto e também com a base de dados *SQLite* na fase de testes [26].

2.5 Conclusões

Este capítulo apresentou as tecnologias e ferramentas fundamentais utilizadas no desenvolvimento da aplicação. A seleção destas tecnologias permitiu garantir uma implementação eficiente e estruturada, tanto no *Back-End* como no *Front-End*. Com esta base, é possível avançar para a descrição da arquitetura e das funcionalidades da aplicação nos próximos capítulos.

Análise de Requisitos e Modelação da Base de Dados

3.1 Introdução

Este capítulo apresenta a análise de requisitos efetuada e o desenvolvimento da base de dados do sistema. Está dividido em três secções principais:

- **Requisitos Funcionais** 3.2, onde são identificadas as principais funcionalidades do sistema;
- **Requisitos Não Funcionais** 3.3, onde são identificadas as restrições técnicas que orientaram o desenvolvimento do sistema;
- **Diagrama da Base de Dados** 3.4, onde é descrita a estrutura das tabelas e as relações entre as diferentes entidades do sistema;

O objetivo principal é garantir que a arquitetura da plataforma está alinhada com as necessidades dos utilizadores e as especificações técnicas, permitindo um funcionamento eficiente e consistente do sistema.

3.2 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve disponibilizar aos utilizadores. Os requisitos planeados foram os seguintes:

Tabela 3.1: Requisitos Funcionais do Sistema

ID	Descrição
RF01	Registo e autenticação de utilizadores
RF02	Edição de perfil e eliminação de conta
RF03	Visualização da lista de cafés e respetivos detalhes
RF04	Criação, edição, remoção e visualização de reservas
RF05	Participação em grupos de reservas já existentes
RF06	Processo de compra de jogos com geração de fatura e envio por email
RF07	Sistema de pagamentos integrado com <i>Stripe</i>
RF08	Notificações por email (confirmação de conta, 2FA, faturas)
RF09	Geração de <i>QR Codes</i> associados a compras
RF10	Painel de Gestão para criação e edição do café
RF11	Painel de Gestão para gerir mesas disponíveis nos cafés
RF12	Painel de Gestão para gerir jogos disponíveis nos cafés
RF13	Painel de Administração para gerir cargos dos Utilizadores

3.3 Requisitos Não Funcionais

Os requisitos não funcionais dizem respeito à qualidade do sistema e às restrições técnicas. Os mais relevantes foram:

Tabela 3.2: Requisitos Não Funcionais do Sistema

ID	Descrição
RNF01	Separação entre o <i>Front-End</i> e o <i>Back-End</i> , com comunicação via API REST
RNF02	Interface intuitiva para computador e telemóvel
RNF03	Armazenamento seguro de palavras-passe (utilização de <i>hashing</i> com <i>bcrypt</i>)
RNF04	Segurança na autenticação com suporte a 2FA e JWT
RNF05	Utilização de <i>cookies</i> com regras de segurança (<i>httpOnly</i> e <i>sameSite</i>)
RNF06	Utilização de base de dados relacional <i>MariaDB</i>
RNF07	Suporte a tarefas agendadas no servidor (verificação de reservas expiradas)
RNF08	Documentação automática da API com <i>Swagger</i>
RNF09	Estrutura do código pensada para facilitar a manutenção futura e permitir a escalabilidade do sistema

3.4 Diagrama da Base de Dados

A base de dados começou a ser desenvolvida com um planeamento cuidadoso, tendo em conta os requisitos funcionais do sistema e as relações entre as entidades principais, como utilizadores, cafés, reservas, jogos e mesas. Este planeamento visou garantir a integridade, coerência e eficiência no armazenamento e recuperação dos dados, bem como suportar as funcionalidades essenciais da aplicação, tais como a gestão de reservas e a associação de utilizadores a grupos.

O diagrama da base de dados (ver figura 3.1) foi gerado automaticamente com o *software DBSchema* [27], a partir do esquema definido com o ORM *Sequelize*.

Este diagrama permite visualizar de forma clara as tabelas, os campos principais e as relações entre entidades como utilizadores, cafés, reservas, jogos e mesas. É importante para facilitar a validação da estrutura da base de dados e a comunicação ao longo do desenvolvimento.

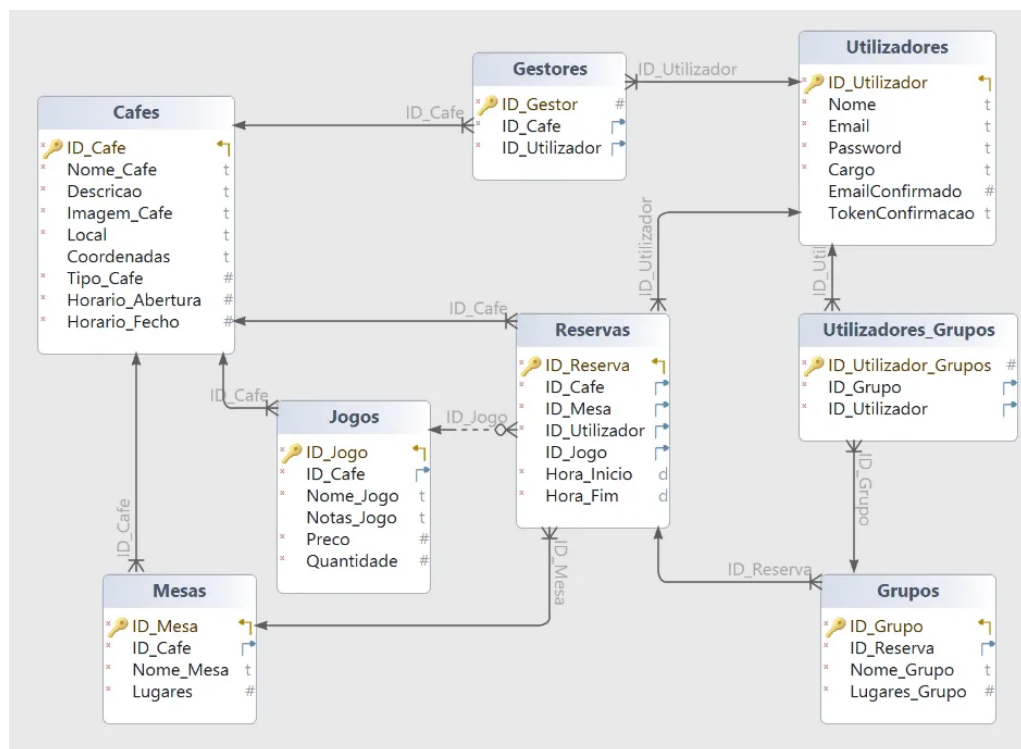


Figura 3.1: Diagrama da Base de Dados criado no *DBSchema*

3.4.1 Estrutura e Relacionamento das Tabelas

O sistema foi modelado com um conjunto de tabelas que representam os principais elementos da aplicação: utilizadores, cafés, jogos, reservas e grupos. As relações definidas entre estas entidades asseguram a integridade e coerência dos dados ao longo do funcionamento do sistema.

As principais tabelas podem ser resumidas da seguinte forma:

- **Utilizadores:** Tabela central com dados dos utilizadores registados, incluindo o estado de confirmação da conta e o cargo do utilizador (utilizador, gestor ou administrador).
- **Cafés:** Contém as informações dos cafés disponíveis, como nome, localização, tipo e horários de funcionamento.
- **Gestores:** Faz a ligação entre os utilizadores e cafés que estes gerem.
- **Mesas:** Cada café possui várias mesas, estando cada uma associada a um número definido de lugares disponíveis e o seu devido nome.
- **Jogos:** Cada café tem os seus próprios jogos, com nome, preço e quantidade.
- **Reservas:** Regista as reservas feitas pelos utilizadores, associando um café, uma mesa, um jogo e um horário.
- **Grupos:** Permite que uma reserva seja partilhada por vários utilizadores.
- **Utilizadores_Grupos:** Relação que associa os utilizadores aos grupos em que participam.

Esta base de dados relacional suporta as principais funcionalidades da aplicação, desde a gestão de cafés e reservas até à criação de grupos para partilhar mesas e jogos.

3.5 Conclusões

A definição clara dos requisitos funcionais e não funcionais permitiu orientar o desenvolvimento de forma estruturada, enquanto a estrutura da base de dados implementada suportou todas as funcionalidades planeadas, mantendo a integridade e coerência dos dados ao longo de toda a aplicação.

Implementação e Testes

4.1 Introdução

Este capítulo descreve as principais funcionalidades implementadas no sistema, bem como os detalhes técnicos relevantes sobre a sua concretização. Está dividido em quatro secções principais:

- **Funcionalidades do Utilizador** 4.2, onde são apresentadas as ações disponíveis para utilizadores comuns da aplicação, como o registo/autenticação, reservas e compras;
- **Funcionalidades de Gestão e Administração** 4.3 que inclui as operações disponíveis para gestores de cafés e administradores da plataforma;
- **Implementação Técnica do *Back-End* e *Front-End*** 4.4, onde se detalha a estrutura do *Back-End* e do *Front-End*, incluindo os controladores, modelos, rotas e componentes;
- **Testes e Validação** 4.5, que descreve os testes efetuados para garantir o correto funcionamento das funcionalidades desenvolvidas.

Cada uma destas secções descreve não só as funcionalidades do ponto de vista do utilizador, mas também os aspetos técnicos relevantes da sua implementação.

4.2 Funcionalidades do Utilizador

Nesta secção é descrita as principais funcionalidades disponibilizadas aos utilizadores registados da aplicação. Estas funcionalidades foram desenvolvidas

com o objetivo de proporcionar uma experiência intuitiva, completa e segura na interação com a aplicação.

4.2.1 Registo e *Autenticação*

Os utilizadores podem criar uma conta através de um formulário de registo simples inserindo o Email, nome e uma palavra-passe. Após o registo, o sistema envia um email de confirmação com um link único para ativar a conta. A *autenticação* é feita com email e palavra-passe, sendo os dados verificados no servidor através de *tokens* JWT.

Existe ainda suporte para 2FA que adiciona uma camada adicional de segurança ao processo de *autenticação*.

A Figura 4.1 apresenta o formulário de registo da aplicação, onde o utilizador insere os seus dados pessoais para criação de conta.

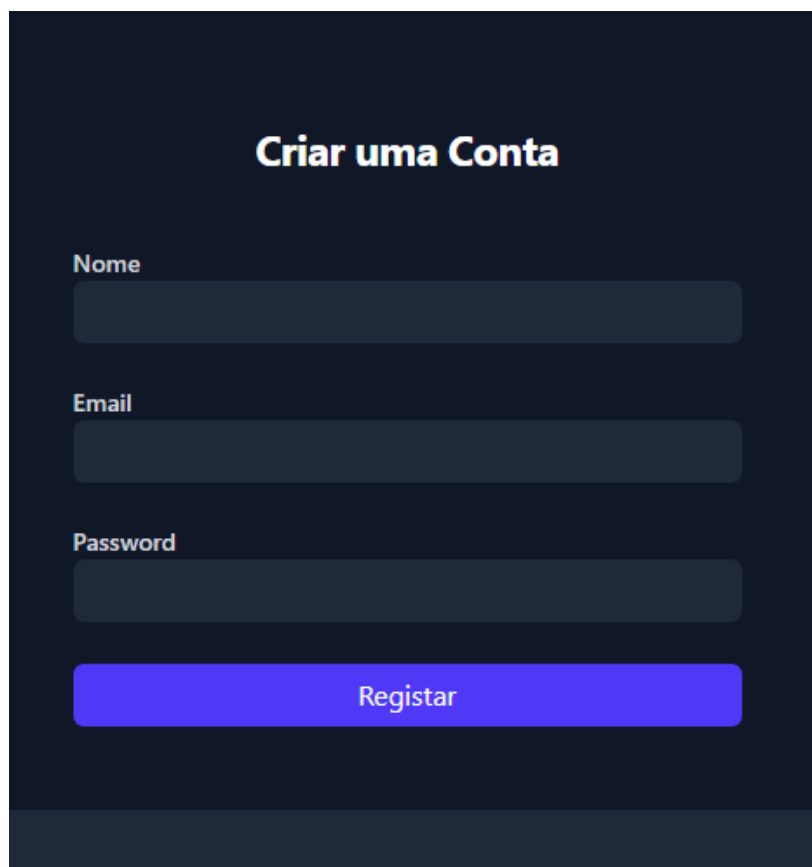
A imagem mostra um formulário de registo com o título "Criar uma Conta" em branco sobre um fundo escuro. Abaixo do título, há três campos de entrada: "Nome", "Email" e "Password", cada um com um rótulo em branco à esquerda e um campo de texto cinza escuro à direita. Abaixo dos campos, há um botão "Registar" em um retângulo amarelo vibrante. O formulário está centralizado na tela.

Figura 4.1: Formulário de Registo de Novo Utilizador

4.2.2 Gestão do Perfil

Após a *autenticação*, o utilizador tem acesso à sua área de perfil, onde pode consultar e editar os seus dados pessoais (nome e palavra-passe), podendo também eliminar a própria conta caso pretenda.

A Figura 4.2 apresenta a interface da área de perfil do utilizador, com opções para alteração de dados e eliminação da conta.

Meu Perfil

Informações Pessoais

Nome: Joao Carneiro

Email: joaomiko257@gmail.com

Cargo: Utilizador

Cancelar

Nome

Nova Senha

Salvar Alterações

Apagar Conta

Minhas Reservas

Reservas em Grupos

Figura 4.2: Área de Gestão de Perfil do Utilizador

4.2.3 Procurar Cafés

A aplicação permite a navegação por uma lista de cafés disponíveis. Ao seleccionar um café, é apresentada uma página com detalhes como a localização, tipo, jogos disponíveis, horários e mesas disponíveis.

A Figura 4.3 ilustra essa lista, mostrando informações básicas de cada café, como nome, tipo, localização e horário.

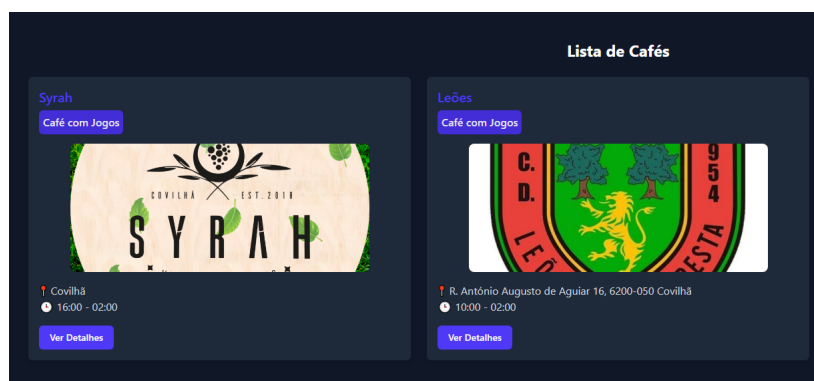


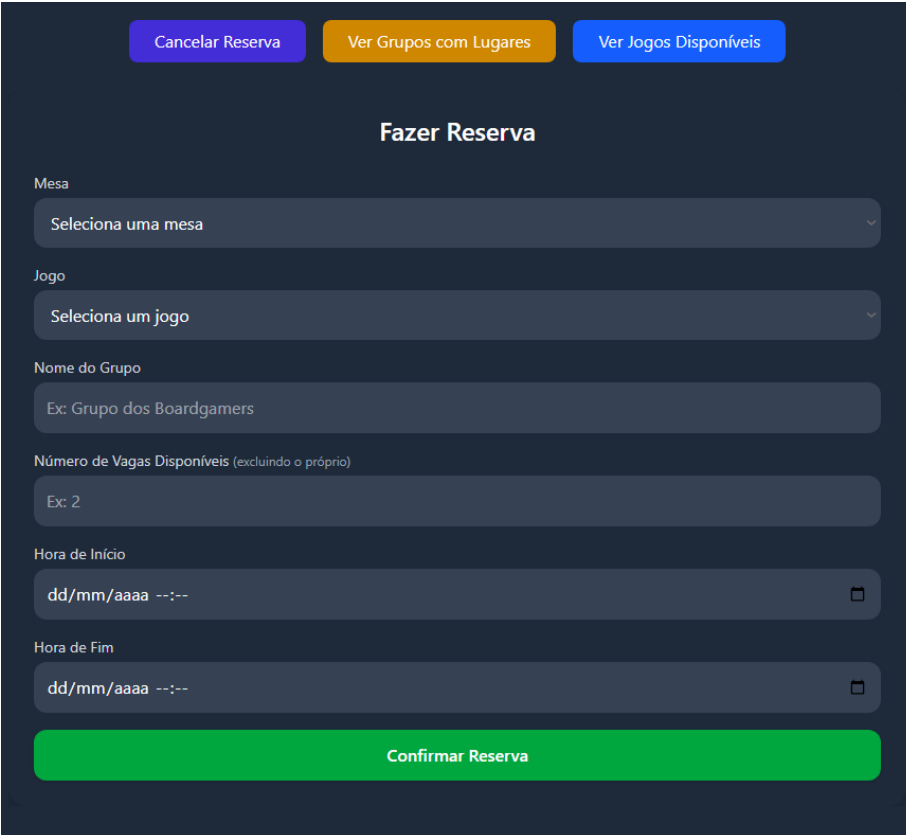
Figura 4.3: Lista de Cafés Disponíveis na Plataforma

4.2.4 Reservas de Mesas e Jogos

Após a selecção de um café, o utilizador pode efetuar uma reserva, escolhendo um horário e a mesa pretendida (que esteja disponível). Para além disso, pode escolher o nome do grupo, o número de lugares disponíveis do grupo e, opcionalmente, um jogo.

Também é possível consultar as reservas ativas no perfil do utilizador, editar (alterar o horário ou número de lugares disponíveis do grupo), ou eliminá-las.

Na Figura 4.4 pode observar-se o formulário para criar uma reserva.



The image shows a web form titled "Fazer Reserva" (Make Reservation) on a dark background. At the top, there are three buttons: "Cancelar Reserva" (Cancel Reservation) in purple, "Ver Grupos com Lugares" (View Groups with Seats) in orange, and "Ver Jogos Disponíveis" (View Available Games) in blue. The form itself has a title "Fazer Reserva" in white. Below the title, there are several input fields: "Mesa" (Table) with a dropdown menu showing "Selecione uma mesa"; "Jogo" (Game) with a dropdown menu showing "Selecione um jogo"; "Nome do Grupo" (Group Name) with a text input field showing "Ex: Grupo dos Boardgamers"; "Número de Vagas Disponíveis (excluindo o próprio)" (Number of Available Seats (excluding self)) with a text input field showing "Ex: 2"; "Hora de Início" (Start Time) with a date and time picker showing "dd/mm/aaaa --:--"; and "Hora de Fim" (End Time) with a date and time picker showing "dd/mm/aaaa --:--". At the bottom of the form is a large green button labeled "Confirmar Reserva" (Confirm Reservation).

Figura 4.4: Criação de Nova Reserva

4.2.5 Juntar-se a Grupos de Reservas

Caso uma mesa ainda tenha lugares disponíveis, outros utilizadores podem juntar-se a essa reserva, formando um grupo. O número de lugares é controlado automaticamente pelo sistema para garantir que não há sobrelotação.

A Figura 4.5 mostra a opção de um utilizador juntar-se a um grupo existente consoante a disponibilidade de lugares disponíveis.



Figura 4.5: Opção para Juntar-se a um Grupo de Reserva

4.2.6 Compras de Jogos

Os utilizadores podem adquirir jogos disponíveis nos cafés. O processo inclui a seleção do jogo, pagamento via *Stripe* e receção da fatura por email. Cada compra gera também um *QR Code* associado, que pode ser utilizado no local para levantamento.

A Figura 4.6 mostra o processo de compra de jogos, com visualização dos jogos e botão de compra integrado com *Stripe*.

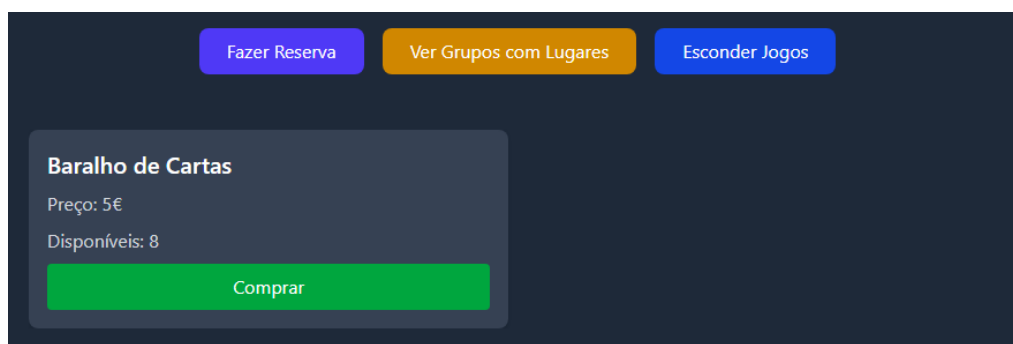


Figura 4.6: Interface de Compra de Jogos

4.3 Funcionalidades de Gestão e Administração

Além das funcionalidades destinadas ao utilizador final, a aplicação inclui interfaces específicas para gestores de cafés e administradores da plataforma. Estas permitem uma gestão eficiente dos recursos e do funcionamento geral do sistema.

4.3.1 Gestão de Cafés

Os gestores autorizados podem aceder a um painel onde podem criar novos cafés, editar os dados de cafés existentes (nome, tipo, localização, horário de funcionamento) ou eliminar cafés que já não estejam ativos. Importa referir que cada café só pode ter um único gestor associado.

A Figura 4.7 apresenta a interface de gestão do café, com opção para editar o café ou eliminar.

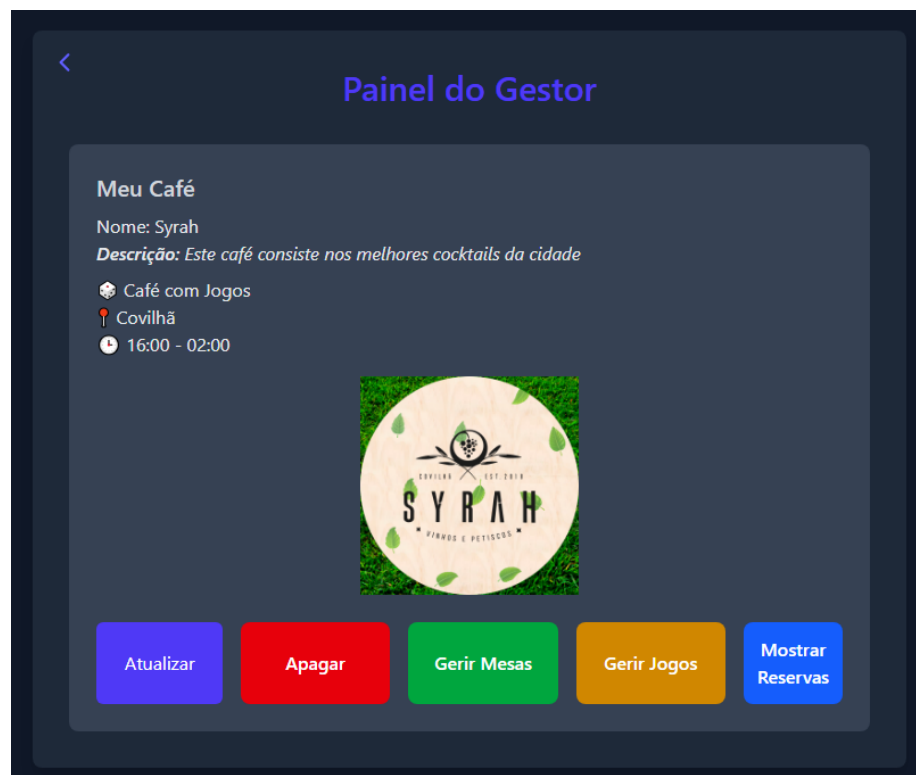
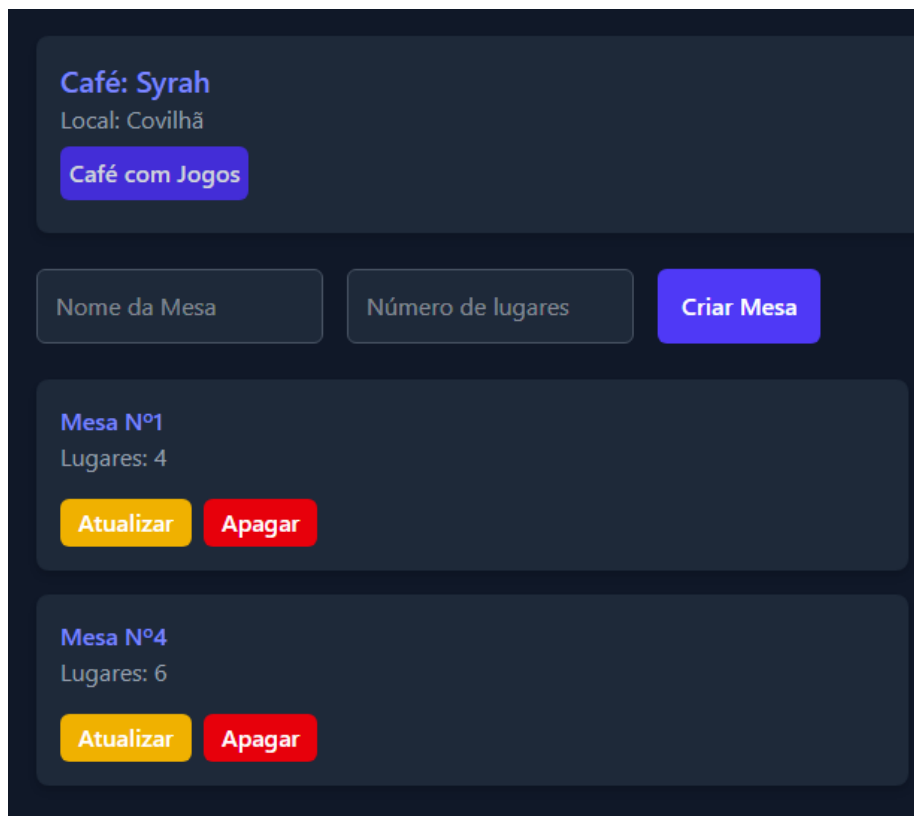


Figura 4.7: Painel de Gestão do Café

4.3.2 Gestão de Mesas

Cada café pode ter associadas várias mesas, com capacidade definida. Os gestores podem adicionar novas mesas, editar o número de lugares disponíveis por mesa ou remover mesas conforme necessário.

A Figura 4.8 mostra a secção dedicada à gestão de mesas, permitindo organizar o nome da mesa e a capacidade de cada uma.



A interface de gestão de mesas para o 'Café: Syrah' no local 'Covilhã'. No topo, há um botão 'Café com Jogos'. Abaixo, há campos para 'Nome da Mesa' e 'Número de lugares', seguidos por um botão 'Criar Mesa'. A interface exibe duas mesas existentes: 'Mesa Nº1' com 4 lugares e 'Mesa Nº4' com 6 lugares. Cada entrada de mesa possui botões 'Atualizar' (amarelo) e 'Apagar' (vermelho).

Café: Syrah
Local: Covilhã
Café com Jogos

Nome da Mesa Número de lugares Criar Mesa

Mesa Nº1
Lugares: 4
Atualizar Apagar

Mesa Nº4
Lugares: 6
Atualizar Apagar

Figura 4.8: Interface de Gestão de Mesas Disponíveis

4.3.3 Gestão de Jogos

Os jogos disponibilizados em cada café também são geridos pelos respetivos gestores. É possível adicionar novos jogos, definir o preço e quantidade em stock, editar as informações existentes ou remover jogos que já não estejam disponíveis.

A Figura 4.9 apresenta a interface de gestão de jogos, onde os gestores podem controlar o *stock* do seu café.

The screenshot shows a dark-themed user interface for managing games. At the top, it identifies the location as 'Café: Syrah' with the local 'Covilhã' and includes a 'Café com Jogos' button. Below this is a 'Criar Novo Jogo' (Create New Game) section with four input fields: 'Nome do Jogo', 'Notas do Jogo', 'Preço', and 'Quantidade', followed by a large blue 'Criar Jogo' button. At the bottom, a 'Baralho de Cartas' (Card Deck) entry is shown with its details: 'Notas:', 'Preço: 5€', and 'Quantidade: 7'. This entry has two action buttons: a yellow 'Atualizar' (Update) button and a red 'Apagar' (Delete) button.

Figura 4.9: Interface de Gestão de Jogos Disponíveis

4.3.4 Gestão de Reservas

Os gestores têm acesso a todas as reservas feitas no seu café, podendo visualizar os detalhes e o estado atual de cada uma. Isto permite uma melhor organização interna e acompanhamento da utilização das mesas e jogos.

A Figura 4.10 apresenta o painel de reservas para gestores, com informações detalhadas sobre cada reserva com o seu devido grupo.

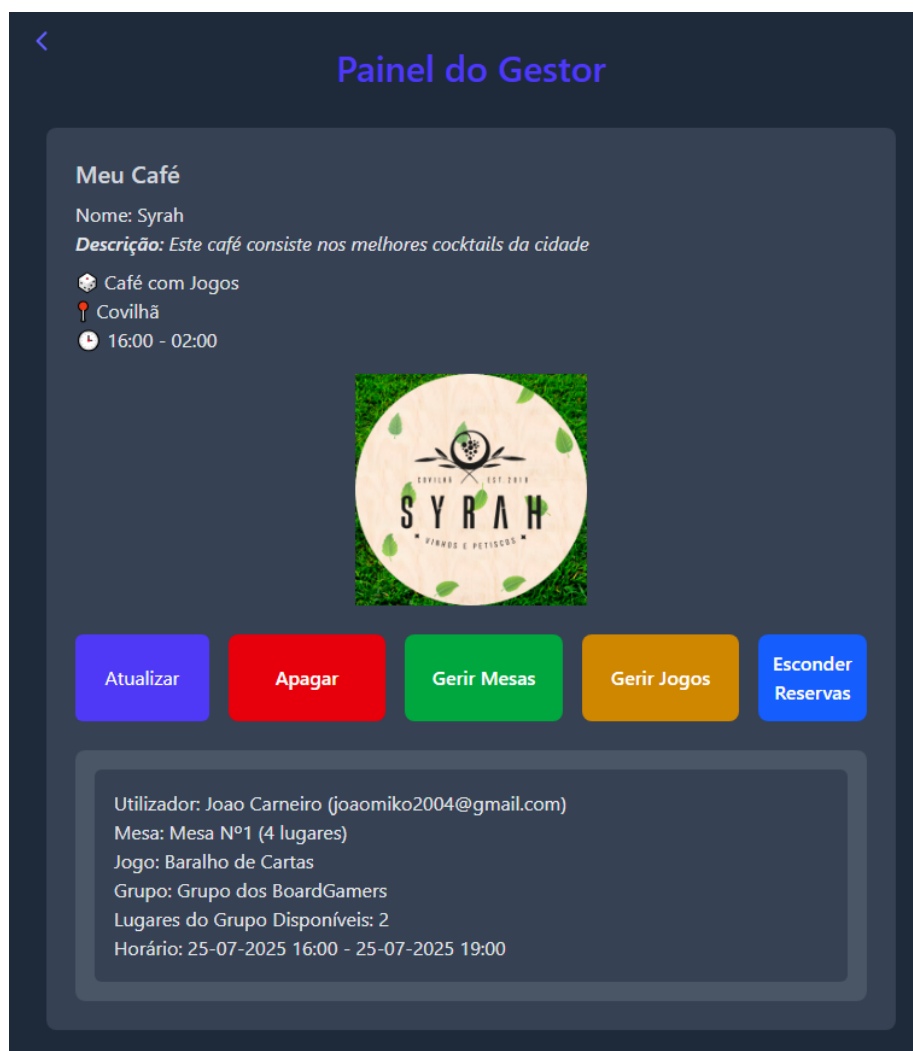


Figura 4.10: Interface de Gestão de Reservas

4.3.5 Gestão de Utilizadores e Cargos (Administração)

A nível administrativo, os utilizadores com privilégios de administrador têm acesso a uma interface para gestão de utilizadores. Nesta, podem atribuir ou remover cargos (utilizador e gestor). Na aplicação, existe apenas um utilizador com permissões de administrador, garantindo um controlo centralizado e seguro sobre as permissões de acesso.

A Figura 4.11 mostra a interface de administração, com a lista de utilizadores e os seus respetivos cargos.

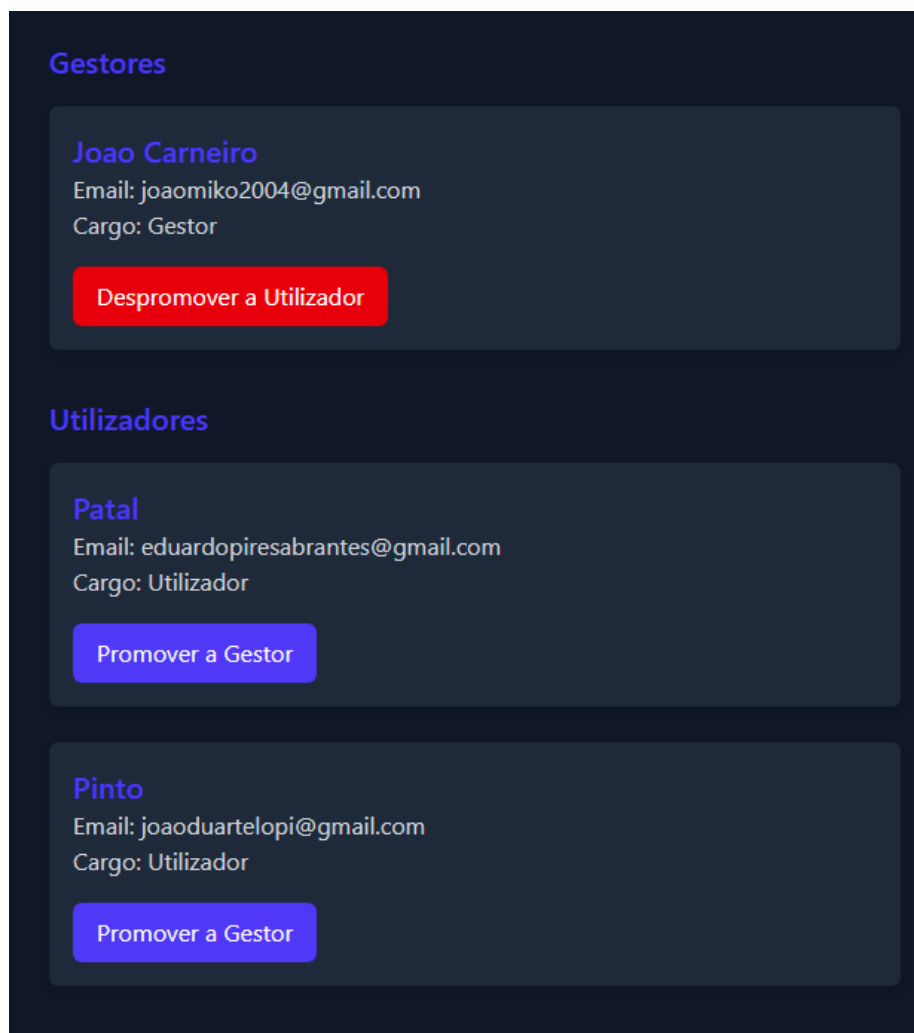


Figura 4.11: Interface de Administração de Utilizadores e Cargos

4.4 Implementação Técnica do *Back-End* e *Front-End*

A implementação do sistema dividiu-se em duas partes principais: o *Back-End*, que trata da lógica da aplicação, gestão dos dados e *autenticação*; e o *Front-End*, que permite a interação do utilizador com o sistema e comunica com a API.

4.4.1 *Back-End*

O *Back-End* da aplicação foi desenvolvido em *Node.js*, utilizando o *framework Express*. Para organizar a estrutura do projeto e promover a escalabilidade, foram criadas pastas separadas para os modelos (*models*), controladores (*controllers*) e rotas (*routes*).

- **Base de Dados (*Database*):** Utiliza o sistema relacional *MariaDB* para armazenar os dados da aplicação. A comunicação é feita através do ORM *Sequelize*, que converte operações em código para instruções *SQL*, garantindo a persistência e integridade da informação.
- **Modelos (*Models*):** Representam as entidades da base de dados utilizando o ORM *Sequelize*. Incluem as definições de atributos, tipos de dados e relações entre tabelas.
- **Controladores (*Controllers*):** Responsáveis por implementar a lógica de cada funcionalidade (ex: criar reserva, *autenticar* utilizador, comprar jogo).
- **Rotas (*Routes*):** Definem os *endpoints* da API REST e ligam-nos aos controladores correspondentes. Algumas rotas estão protegidas com um *middleware* de *autenticação*, só permitindo acesso caso o *token* JWT do utilizador seja válido.
- **Middlewares:** Foram implementados *middlewares* para autenticação JWT, controlo de permissões (administrador/gestor/utilizador), validação de dados e tratamento de erros.

Além disso, foram incluídas funcionalidades adicionais como:

- **Criação e envio de *emails*:** Utilizando a biblioteca *Nodemailer*, é possível enviar emails para confirmação de conta, 2FA e envio de faturas com *QR Code*. A seguir apresenta-se um excerto da função de envio de faturas por email:

```
const qrContent = `Compra: ${nomeJogo} | Cliente: ${
  nomeCliente}`;
const qrDataURL = await QRCode.toDataURL(qrContent);
const doc = new PDFDocument();
doc.text(`Jogo: ${nomeJogo}`);
doc.image(qrBuffer, { width: 120 });
transporter.sendMail({
  to: destinatario,
  subject: 'Obrigado pela tua compra!',
  attachments: [{ filename: 'fatura.pdf', path: pdfPath
  }]
});
```

Excerto de Código 4.1: Envio de fatura com QR Code via email

- **Agendamento de tarefas periódicas:** Com a biblioteca *node-cron*, o sistema verifica reservas expiradas e repõe o stock de jogos automaticamente. Abaixo mostra-se um exemplo desta verificação executada a cada minuto:

```
cron.schedule('* * * * *', async () => {
  const reservasExpiradas = await Reservas.findAll({
    where: { Hora_Fim: { [Op.lte]: new Date() } }
  });
  for (const reserva of reservasExpiradas) {
    const jogo = await Jogos.findByPk(reserva.ID_Jogo);
    if (jogo) jogo.Quantidade += 1;
    await reserva.destroy();
  }
});
```

Excerto de Código 4.2: Tarefa agendada para atualizar reservas e stock

- **Integração com Stripe para pagamentos:** Para processar pagamentos de jogos, foi utilizada a biblioteca oficial da *Stripe*. A criação da sessão de pagamento pode ser observada no excerto seguinte:

```
const session = await stripe.checkout.sessions.create({
  payment_method_types: ['card'],
  customer_email: utilizador.Email,
  line_items: [{
    price_data: {
      currency: 'eur',
      product_data: { name: jogo.Nome_Jogo },
      unit_amount: Math.round(jogo.Preco * 100)
    },
    quantity: 1
  }
});
```

```
    }],  
    success_url: `${process.env.CLIENT_URL}/cafe/sucesso?  
        session_id={CHECKOUT_SESSION_ID}`,  
    cancel_url: `${process.env.CLIENT_URL}/perfil`  
  });
```

Excerto de Código 4.3: Criação de sessão de pagamento com Stripe

- **Documentação automática da API:** A API foi documentada com recurso ao *Swagger*, permitindo aos programadores testar os endpoints diretamente a partir da interface web. Esta documentação é gerada automaticamente com base nas anotações dos controladores e rotas. <https://diceandtables.pt/api-docs/>

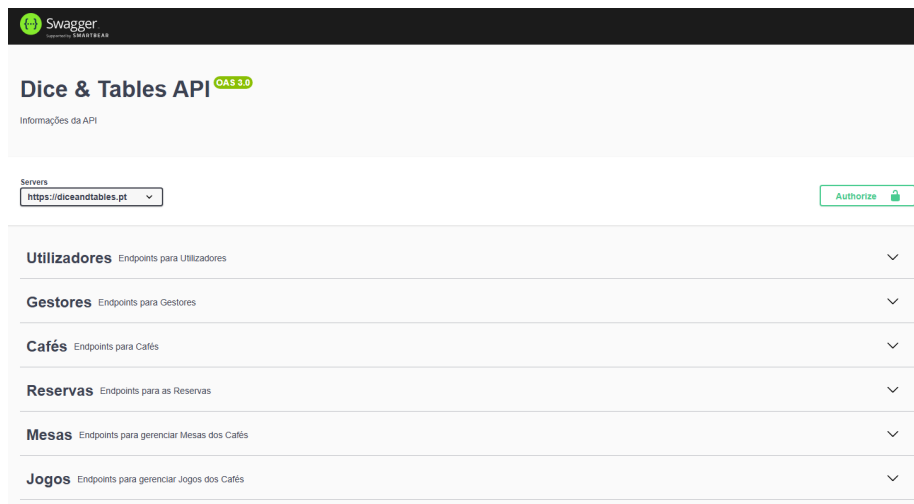


Figura 4.12: Documentação automática da API com Swagger

4.4.2 Front-End

O *Front-End* foi desenvolvido com *React*, usando o *Vite* como ambiente de desenvolvimento. A interface está dividida em componentes reutilizáveis e organizados por secções.

- **Gestão de Estado e Dados:** Utilizou-se a biblioteca *TanStack React Query* para a gestão da cache e comunicação com a API.
- **Routing:** A navegação entre páginas foi implementada com *TanStack Router*, permitindo rotas aninhadas e carregamento dinâmico de dados.

- **Autenticação:** O *token* JWT é armazenado de forma segura em *cookies* HTTP-only, com o apoio da biblioteca *js-cookie*.
- **Notificações e Feedback:** A biblioteca *React Toastify* foi usada para apresentar mensagens de sucesso, erro ou informação ao utilizador.
- **Pagamentos:** A biblioteca *@stripe/stripe-js* permite criar e confirmar pagamentos diretamente no cliente, de forma segura.

Ambas as partes comunicam entre si através de uma API REST bem definida, garantindo uma separação clara de responsabilidades entre o *Front-End* e o *Back-End*.

A estrutura de pastas do *Front-End* foi organizada para garantir clareza e modularidade. No diretório principal *src/* destacam-se as seguintes subpastas:

- *assets/*: Contém recursos estáticos como imagens, ícones e ficheiros CSS globais.
- *components/*: Agrupa componentes reutilizáveis da interface, responsáveis por pequenas unidades funcionais e visuais.
- *context/*: Inclui os *context providers* do React usados para gestão de estado global e partilhado entre componentes.
- *routes/*: Define a organização das rotas da aplicação, com subpastas específicas, como *cafe/*, que contém componentes e páginas relacionadas à funcionalidade de cafés.

Esta organização permite manter a aplicação escalável e facilita a manutenção e expansão futura.

4.5 Testes e Validação

A fase de testes teve como objetivo validar o funcionamento do sistema e garantir que os requisitos definidos foram cumpridos.

4.5.1 Testes de Funcionalidade

A validação funcional foi realizada através de testes manuais que garantiram o correto funcionamento de cada requisito identificado. A Tabela 4.1 apresenta um resumo dos testes realizados e os respetivos resultados obtidos.

Tabela 4.1: Validação dos Requisitos Funcionais

ID	Teste Realizado	Resultado Obtido
RF01	Registar um novo utilizador e fazer <i>login</i>	O registo foi bem-sucedido. O utilizador recebeu um email de confirmação e conseguiu iniciar sessão após a validação.
RF02	Acéder ao perfil, editar dados e eliminar conta	A edição de perfil foi bem-sucedida. A eliminação da conta removeu todos os dados associados.
RF03	Acéder à lista de cafés e aos detalhes de cada um	A lista foi carregada corretamente. Os detalhes de cada café apresentaram localização, tipo e horários.
RF04	Criar, editar e remover uma reserva	A reserva foi criada com sucesso. As alterações foram persistidas e a remoção foi refletida na base de dados.
RF05	Juntar-se a uma reserva com lugares disponíveis	O sistema detetou corretamente os lugares disponíveis e permitiu a associação do utilizador ao grupo.
RF06	Comprar um jogo, gerar fatura e receber por email	A compra foi concluída. Uma fatura em PDF foi gerada e enviada para o email do utilizador.
RF07	Efetuar pagamento via <i>Stripe</i>	O pagamento foi processado com sucesso, utilizando o ambiente de teste da plataforma.
RF08	Validar envio de <i>emails</i> (confirmação, 2FA, faturas)	Todos os <i>emails</i> foram enviados corretamente, com os conteúdos esperados.
RF09	Validar geração e leitura de <i>QR Codes</i>	O <i>QR Code</i> foi gerado após a compra e pôde ser lido por leitores comuns, contendo os dados esperados.
RF10	Criar e editar cafés no <i>Painel de Gestão</i>	Os cafés foram criados e editados com sucesso. As alterações refletiram-se na interface e na base de dados.
RF11	Gerir mesas no painel do gestor	Foi possível adicionar, editar e remover mesas, com número de lugares e nomes validados.
RF12	Gerir jogos disponíveis num café	Os jogos foram corretamente associados ao café. A edição e remoção funcionaram como esperado.
RF13	Alterar cargos de utilizadores no painel de administração	Foi possível promover utilizadores a gestores. As permissões mudaram consoante o cargo.

4.5.2 Testes de Integração

Foram testadas as interações entre os vários componentes do sistema, com especial foco na comunicação e troca de dados entre os módulos. Estes testes permitiram validar que:

- A comunicação entre o *Front-End* e o *Back-End*, realizada através de chamadas à API REST, ocorre de forma estável e sem erros, com as respostas apropriadas a diferentes estados e inputs;
- O armazenamento e a recuperação de dados na base de dados funcionam corretamente, com operações de *Create*, *Read*, *Update* and *Delete*.

lete (CRUD) bem-sucedidas para todas as entidades principais do sistema;

- A integração com serviços externos, como *Stripe* para pagamentos, *No-demailer* para envio de emails e *QRCode* para geração de códigos, decorre sem falhas e com os dados esperados;
- As tarefas agendadas no servidor, como a remoção automática de reservas expiradas, são executadas corretamente segundo a frequência definida.

4.5.3 Validação Manual

A validação foi realizada de forma iterativa durante o desenvolvimento, recorrendo a diferentes perfis de utilizador para simular cenários reais e identificar eventuais falhas.

4.6 Conclusões

Ao longo deste capítulo foram descritos os vários pormenores da implementação da aplicação que permitiu alcançar os objetivos definidos, assegurando uma aplicação funcional, segura e preparada para uma utilização real. As funcionalidades estão organizadas consoante os diferentes perfis de utilizador, e a comunicação entre componentes foi validada com sucesso. A estrutura modular do código facilita a sua manutenção e expansão futura.

Conclusões, Dificuldades Encontradas e Trabalho Futuro

5.1 Principais Conclusões

Com o desenvolvimento deste projeto, foram aplicados e consolidados conhecimentos em desenvolvimento *web*, modelação de bases de dados e engenharia de *software*. A plataforma desenvolvida responde à necessidade de encontrar cafés com jogos de tabuleiro, reservar mesas e organizar grupos de jogo.

Foi implementado um *Back-End* em *Node.js* e um *Front-End* em *React*, com autenticação segura, gestão de utilizadores e integração com serviços externos como *Stripe* e envio de *emails* com *Nodemailer*. Também foi colocado o projeto online que pode ser acedido no seguinte domínio: <https://diceandtables.pt>.

No final, considera-se que a plataforma cumpre os objetivos definidos, demonstrando a utilidade prática dos conhecimentos adquiridos ao longo da licenciatura e abrindo portas a futuras melhorias e expansões.

5.2 Dificuldades Encontradas

Durante o desenvolvimento do projeto, surgiram várias dificuldades técnicas e de integração que exigiram pesquisa, experimentação e reformulação de abordagens. As principais foram:

- **Validação de reservas e sobreposição de horários:** foi necessário implementar uma lógica rigorosa para garantir que não existissem conflu-

tos de horários entre reservas, considerando a disponibilidade de mesas e os horários de funcionamento dos cafés. A resolução envolveu a análise detalhada das datas com operadores do *Sequelize* e testes em diversos cenários.

- **Gestão de horários que atravessam a meia-noite:** um outro desafio, também relacionado com horários, foi no tratamento de reservas que começam num dia e terminam no seguinte (por exemplo, das 23h às 01h). Foi necessário adaptar a lógica para que o sistema reconhecesse corretamente estas situações como válidas, sem as considerar como inválidas ou sobrepostas.
- **Gestão de relações entre entidades:** definir corretamente as associações entre utilizadores, grupos, reservas e cafés exigiu um planeamento cuidadoso da estrutura da base de dados e a utilização correta das funcionalidades do ORM *Sequelize*, como *hasMany* e *belongsToMany*.
- **Integração com serviços externos:** a integração com o *Stripe* para pagamentos e com o *Nodemailer* para envio de emails apresentou desafios relacionados com autenticação, tratamento de erros e formatação de conteúdos dinâmicos, que foram ultrapassados com base na documentação oficial e testes incrementais.

A superação destas dificuldades contribuiu para o reforço das minhas competências técnicas e para uma maior preparação para lidar com problemas reais em ambientes de desenvolvimento profissional.

5.3 Trabalho Futuro

Embora os objetivos principais tenham sido cumpridos, inclusive algumas sugestões e melhorias que não estavam planeadas mas que foram surgindo ao longo do desenvolvimento do projeto, existem várias oportunidades de melhoria. Entre as sugestões de trabalho futuro, destacam-se:

- **Melhorias na interface do utilizador:** embora funcional, a interface poderá beneficiar de um *design* mais moderno, incluindo animações e uma melhor organização visual.
- **Painel estatístico:** seria interessante adicionar gráficos com estatísticas de reservas, utilização de cafés, jogos mais populares e métricas de desempenho do sistema.

- **Sistema de avaliação de cafés e jogos:** permitir aos utilizadores deixar *feedback* e avaliações poderia aumentar o envolvimento com a plataforma.
- **Notificações em tempo real:** através de *WebSockets* ou bibliotecas como *Socket.IO* [28], seria possível notificar os utilizadores sobre alterações nas suas reservas ou interações de grupo em tempo real.
- **Internacionalização da aplicação:** adicionar suporte a múltiplos idiomas para alcançar um público mais vasto.

Estas ideias representam possíveis evoluções do projeto, tanto para fins académicos como comerciais. O sistema desenvolvido pode ser facilmente adaptado para outros contextos semelhantes, como reservas em bibliotecas ou salas de estudo.

Bibliografia

- [1] BBC News. Why board games are booming in the age of ai, 2025. [Online] <https://www.bbc.com/news/articles/clyw3d0qeeko>. Último acesso a 13 de Junho de 2025.
- [2] GlobeNewswire. Tabletop games market growth forecasts 2025–2030, 2025. [Online]. Disponível em: <https://www.globenewswire.com>. Último acesso em 13 jun. 2025.
- [3] React Team. React – A JavaScript library for building user interfaces, 2024. [Online] <https://react.dev/>. Último acesso a 13 de Junho de 2025.
- [4] Node.js Foundation. Node.js Documentation, 2024. [Online] <https://nodejs.org/en/docs>. Último acesso a 13 de Junho de 2025.
- [5] BoardGameGeek. BoardGameGeek – The World's Largest Collection of Board Game Data, 2025. [Online] Disponível em: <https://boardgamegeek.com>. Último acesso em 23 de junho de 2025.
- [6] Meetup, Inc. Meetup – We are what we do, 2025. [Online] Disponível em: <https://www.meetup.com>. Último acesso em 13 de junho de 2025.
- [7] OpenTable, Inc. OpenTable – Restaurant Reservation System, 2025. [Online] <https://www.opentable.com>. Último acesso em 23 de junho de 2025.
- [8] BookingNinja. Bookingninja: Booking system for board game cafes, 2025. [Online] Disponível em: <https://bookingninja.io>. Último acesso em 13 jun. 2025.
- [9] Buse-Ioan. Board-game-cafe-reservation-system: Spring boot reservation system for board game cafés, 2024. [Online] Disponível em: <https://github.com/Buse-Ioan/Board-Game-Cafe-Reservation-System>. Último acesso em 13 jun. 2025.

- [10] Express.js Contributors. Express – Fast, unopinionated, minimalist web framework for Node.js, 2024. [Online] <https://expressjs.com/>. Último acesso a 13 de Junho de 2025.
- [11] Sequelize Contributors. Sequelize – Node.js ORM for relational databases, 2024. [Online] <https://sequelize.org/>. Último acesso a 13 de Junho de 2025.
- [12] MariaDB Foundation. MariaDB – Open source database, 2024. [Online] <https://mariadb.org/>. Último acesso a 13 de Junho de 2025.
- [13] SQLite Team. SQLite – Self-contained, high-reliability, embedded, full-featured, public-domain SQL database engine, 2024. [Online] <https://sqlite.org/>. Último acesso a 13 de Junho de 2025.
- [14] bcrypt.js contributors. bcrypt.js – Optimized bcrypt in JavaScript with zero dependencies, 2024. [Online] <https://github.com/dcodeIO/bcrypt.js>. Último acesso a 13 de Junho de 2025.
- [15] Auth0. jsonwebtoken – JSON Web Token implementation for Node.js, 2024. [Online] <https://github.com/auth0/node-jwebtoken>. Último acesso a 13 de Junho de 2025.
- [16] Stripe, Inc. Stripe – Online payment processing for internet businesses, 2024. [Online] <https://stripe.com/>. Último acesso a 13 de Junho de 2025.
- [17] Tailwind Labs. Tailwind css – a utility-first css framework, 2025. [Online]. Disponível em: <https://tailwindcss.com>. Último acesso em 13 jun. 2025.
- [18] TanStack. TanStack Query – Powerful asynchronous state management for TS/JS, 2024. [Online] <https://tanstack.com/query>. Último acesso a 13 de Junho de 2025.
- [19] Axios Team. Axios – Promise based HTTP client for the browser and node.js, 2024. [Online] <https://axios-http.com/>. Último acesso a 13 de Junho de 2025.
- [20] js-cookie Contributors. js-cookie – A simple, lightweight JavaScript API for handling cookies, 2024. [Online] <https://github.com/js-cookie/js-cookie>. Último acesso a 13 de Junho de 2025.

-
- [21] Fadi Khadra. React Toastify – React notification made easy, 2024. [Online] <https://fkhadra.github.io/react-toastify/>. Último acesso a 13 de Junho de 2025.
 - [22] Stripe, Inc. @stripe/stripe-js, 2024. [Online] <https://github.com/stripe/stripe-js>. Último acesso a 13 de Junho de 2025.
 - [23] Microsoft. Visual Studio Code – Code editing. Redefined., 2024. [Online] <https://code.visualstudio.com/>. Último acesso a 13 de Junho de 2025.
 - [24] Kong Inc. Insomnia – API design platform and REST client, 2024. [Online] <https://insomnia.rest/>. Último acesso a 13 de Junho de 2025.
 - [25] GitHub, Inc. GitHub – Where the world builds software, 2024. [Online] <https://github.com/>. Último acesso a 13 de Junho de 2025.
 - [26] HeidiSQL Project. HeidiSQL – GUI client for MariaDB, MySQL, MS SQL, PostgreSQL, 2024. [Online] <https://www.heidisql.com/>. Último acesso a 13 de Junho de 2025.
 - [27] DBSchema Team. DBSchema – Visual database designer manager, 2024. [Online] <https://dbschema.com/>. Último acesso a 13 de Junho de 2025.
 - [28] Socket.IO Contributors. Socket.IO – Real-time application framework for Node.js, 2024. [Online] <https://socket.io/>. Último acesso a 13 de Junho de 2025.